Motivating Trends in Computer Architecture

Trend 1
Power & energy constrain all systems

Trend 2
Transition to multicore processors

Trend 3
Inevitable end of Moore’s law

Data collected by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, C. Batten
- Batten Research Group Overview
- XLOOPS: Specialization for Loop Patterns

- Design
  - Performance
  - Constraint

- Embedded
  - Architectures

- Simple
  - Processor

- High-Performance
  - Architectures

- Less Flexible
  - Accelerator

- More Flexible
  - Accelerator

- Custom
  - ASIC

- Flexibility vs. Specialization

- Energy Efficiency (Tasks per Joule)
- Performance (Tasks per Second)

Cornell University
Christopher Batten
Fine-Grain Heterogeneous Architectures

Latency-Optimized Tile: few large out-of-order cores

Throughput-Optimized Tile: many small in-order cores

Data-Parallel-Optimized Tile: flexible data-parallel accelerator

Application-Optimized Tile: instruction set specialization; flexible algorithm or data-structure accelerators
Projects Within the Batten Research Group

GPGPU Microarchitecture

XLOOPS: Specialization for Loop Patterns

XPC: Explicit-Parallel-Call Architectures

Polymorphic Hardware Specialization

Reconfigurable Power Distribution Networks

Complexity Effective On-Chip Networks

Hardware Acceleration for Dynamic Languages

Python-Based Hardware Modeling

```
class Reg (Model):
    def __init__( s ):
        s.in  = InPort(32)
        s.out = OutPort(32)
    def elaborate( s ):
        @s.posedge_clk
        def seq_logic():
            s.out.next = s.in
```
Vertically Integrated Research Methodology

Our research involves reconsidering all aspects of the computing stack including applications, programming frameworks, compiler optimizations, runtime systems, instruction set design, microarchitecture design, VLSI implementation, and hardware design methodologies.

Experimenting with full-chip layout, FPGA prototypes, and test chips is a key part of our research methodology.

Key Metrics: Cycle Count, Cycle Time, Area, Energy
XLOOPS: Architectural Specialization for Inter-Iteration Loop Dependence Patterns

Shreesha Srinath, Berkin Ilbeyi, Mingxing Tan, Gai Liu, Zhiru Zhang, and Christopher Batten

Explicit loop specialization (XLOOPS) elegantly encodes inter-iteration loop dependence patterns in the instruction set and enables traditional, specialized, and adaptive execution.
XLOOKS ISA: Unordered Concurrent

\[
\text{for } ( i=0; i<N; i++ ) \quad C[i] = A[i] \times B[i]
\]

\[
\text{loop:}
\]

- \text{lw} \quad r2, 0(rA)
- \text{lw} \quad r3, 0(rB)
- \text{mul} \quad r4, r2, r3
- \text{sw} \quad r4, 0(rC)
- \text{addiu.xi} \quad rA, 4
- \text{addiu.xi} \quad rB, 4
- \text{addiu.xi} \quad rC, 4
- \text{addiu} \quad r1, r1, 1
- \text{xloop.uc} \quad r1, rN, \text{loop}
XLOOPS ISA: Ordered-Through-Registers

for ( X=0, i=0; i<N; i++ )
    X += A[i]
    B[i] = X

loop:
    lw r2, 0(rA)
    addu rX, r2, rX
    sw rX, 0(rB)
    addiu.xi rA, 4
    addiu.xi rB, 4
    addiu r1, r1, 1
    xloop.or r1, rN, loop
XLOOPS ISA: Ordered-Through-Memory

for ( i=K; i<N; i++ )

    # r1 = rK
    # r3 = rA + 4*rK

loop:
    lw r4, 0(r3)
    lw r5, 0(rA)
    mul r6, r4, r5
    sw r6, 0(r3)
    addiu.xi r3, 4
    addiu.xi rA, 4
    addiu r1, r1, 1
    xloop.om r1, rN, loop
XLOOPS ISA: Unordered Atomic

```
for ( i=0; i<N; i++ )
    B[ A[i] ]++
    D[ C[i] ]++

loop:
    lw      r6, 0(rA)
    lw      r7, 0(r6)
    addiu   r7, r7, 1
    sw      r7, 0(r6)
    addiu.xi rA, rA, 4
    ...
    addiu   r1, r1, 1
    xloop.ua r1, rN, loop
```
XLOOPS ISA: Dynamic Bound

Iteration 0
inst0
inst1
inst2
inst3
...
xloop.db

Iteration 1
inst0
inst1
inst2
inst3
...
xloop.db

Iteration 2
inst0
inst1
inst2
inst3
...
xloop.db

Iteration 3
inst0
inst1
inst2
inst3
...
xloop.db

Iteration 4
inst0
inst1
inst2
inst3
...
xloop.db

Iteration 5
inst0
inst1
inst2
inst3
...
xloop.db
Kernel implementing Floyd-Warshall shortest path algorithm

```c
for ( int k = 0; k < n; k++ )
    #pragma xloops ordered
    for ( int i = 0; i < n; i++ )
        #pragma xloops unordered
        for ( int j = 0; j < n; j++ )
            path[i][j] = min( path[i][j], path[i][k] + path[k][j] );
```

Kernel implementing greedy algorithm for maximal matching

```c
#pragma xloops ordered
for ( int i = 0; i < num_edges; i++ )
    int v = edges[i].v; int u = edges[i].u;
    if ( vertices[v] < 0 && vertices[u] < 0 )
        vertices[v] = u; vertices[u] = v; out[k++] = i;
```
XLOOPS Microarchitecture

- Elegant hardware/software abstraction enables efficient traditional execution on general-purpose processor
- Loop-pattern specialization unit enables specialized execution for improved performance and efficiency
- Adaptive execution uses two profiling phases to adaptively determine best location to run loop
XLOOPS Cycle-Level Performance Results

- XLOOPS vs. Simple Core: Always higher performance
- XLOOPS vs. OOO 2-way: Generally competitive or higher performance
- XLOOPS vs. OOO 4-way: Mixed results
XLOOPS Cycle-Level Energy-Efficiency Results

- XLOOPS vs. Simple RISC Core: Competitive energy efficiency
- XLOOPS vs. OOO 2-way: Always more energy efficient
- XLOOPS vs. OOO 4-way: Always more energy efficient
VLSI Implementation

- TSMC 40 nm standard-cell-based implementation
- RISC scalar processor with 4-lane LPSU
- Supports xloop.uc
- Implemented in TSMC 40 nm using ASIC CAD toolflow
- $\approx 40\%$ extra area compared to simple RISC processor
Take-Away Points

- We envision future computing systems using fine-grain heterogeneous architectures with a mix of tiles optimized for latency, throughput, classes of applications, or even a specific application.

- We are using a vertically integrated research methodology to explore a variety of projects that will hopefully enable such fine-grain heterogeneous architectures.